Syniti

Syniti Replicate

Setup Notes for IBM® Informix® Transactional Replications with Triggers

Version 10.2

Syniti

Syniti Replicate

Table of Contents

Syniti Replicate

These notes provide essential information for setting up replications using **IBM Informix**.

This guide describes the setup process using the Triggers option for one-way mirroring and synchronization when replicating data from an IBM Informix database.

For mirroring and synchronization replications using IBM Informix as a source, Syniti Replicate offers two approaches:

- **Log Server Agent:** Uses a Windows service and a Log Server component to query the Informix log for increased performance when dealing with large amounts of data. This approach is described in the document *Setup Notes for Replicating with IBM Informix* available in the Help Center.
- **Triggers:** Uses triggers installed on the Informix database to log changes.

Connection Type

Informix .NET Provider in the latest Informix Client SDK (recommended version 4.10 or above) with the ESQL option.
Assembly: IBM.Data.Informix

Issues with Informix Provider Versions

- There appear to be some issues with later builds/FCs of the Informix 3.70 provider. Syniti recommends using 4.10 or above.
- There can be only one Informix provider installed the system. If other versions of the provider, such as the one installed with IBM DB2, are installed, uninstall them and restart the system.
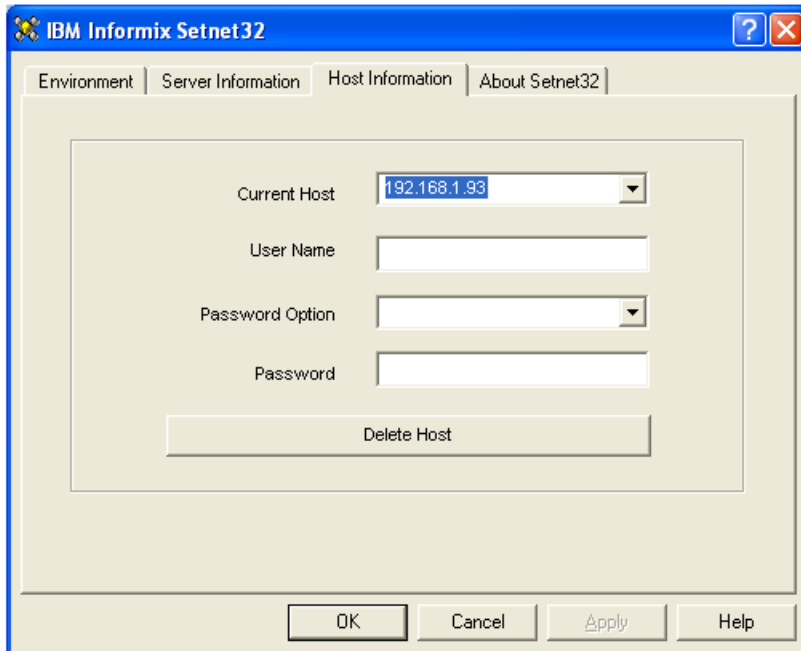
1. Run Setnet32 from All Programs-> IBM Informix Client-SDK X.XX
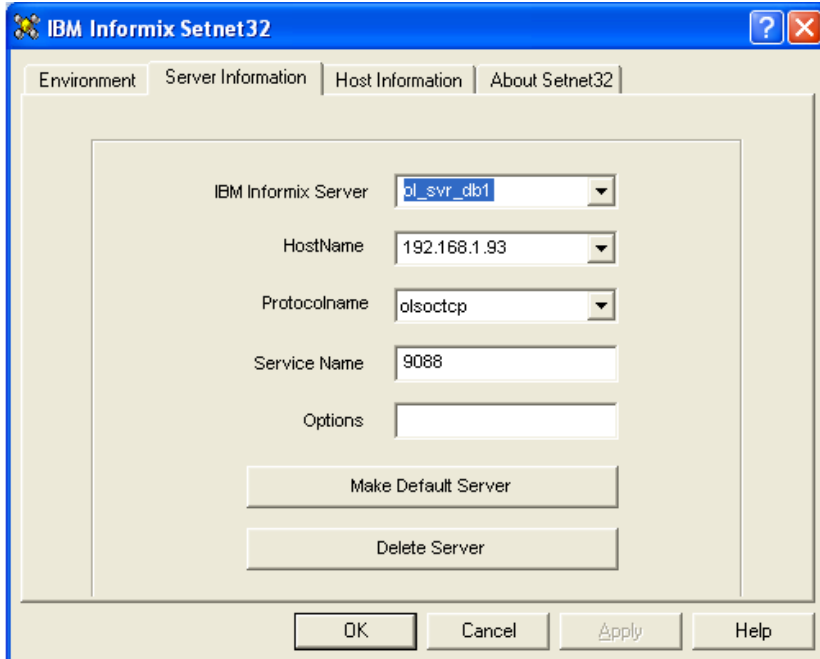
Syniti Replicate

2. Enter the Server Information based on your Informix Login.

3. Click **Apply**.
4. Verify the connection to the database using the ILogin.exe demo. If you are unable to connect, review the issues below to identify errors or to see if you have the correct variables set.

INFORMIXSERVER Windows System Variable

The Informix client software requires a default value to be set for INFORMIXSERVER on the client system (i.e. the system where both Syniti Replicate and the Informix client software are installed.) The INFORMIXSERVER environment variable must specify a default database server.

DELIMIDENT Informix Database Variable Setting

Please check if there are space(s) or special characters in the schema or table name.

The variable DELIMIDENT needs to be set to YES (DELIMIDENT=YES) in the Informix database because Syniti Replicate needs to run the following statement with double quotes on the table name when the replication is created.

CREATE TABLE baan."_DBM__TRG_OBJS" (SNAME VARCHAR(32), TNAME VARCHAR(128), TAG INTEGER, MASTERTBL VARCHAR(128))

If the variable DELIMIDENT is not set, the above query fails with a syntax error.

Informix Error Numbers and Descriptions

To look up an Informix error number: when the Informix client software is installed, a Program Group named IBM Informix Connect (or IBM Informix Client SDK) is created.
The program group includes an application called Find Error, with descriptions of all error numbers.

Syniti Replicate

Trigger-Based Replication Overview

A database trigger is code that is automatically executed in response to certain events on a database table. To define a trigger-based replication (mirroring or synchronization), you need to provide information in the Source and/or Target Connection wizards so that triggers can be created to log table changes for replication.

For each table involved in the replication, Syniti Replicate creates 3 triggers in the source table that fire when a specific event occurs on a record:

- INSERT trigger which fires when a new record is being inserted in the table
- UPDATE trigger which fires when a record is modified
- DELETE trigger which fires when a record is deleted

If the replication is later deleted, the triggers are removed by Syniti Replicate. However, note that if you change a replication from mirroring to refresh, the triggers on the source table are not deleted. All transactions will continue to be recorded in the log tables. If you are not planning to reset the replication to mirroring, it is better to delete the replication, so that the triggers are removed, and create a new refresh replication.

Data retrieved using the triggers is stored in log tables that are specified in your Source/Target connection. The master log table can be an existing table or one created specifically to hold Syniti Replicate information. It contains general information about the transactions, like user name, timestamp, table name. A log table (_DBM__LOG_x) is also created for each source table in the replication, and contains the data changes identified by the triggers, as well as trigger objects _DBM__TRG_OBJS.

Note that Syniti Replicate does not create a tablespace. If you want to have a table space named SYNITIDR, you must create it beforehand using a SQL tool. Run the appropriate SQL statement for your environment. For example:

CREATE TABLESPACE SYNITIDR

When creating a connection, it is important to set the retention time to keep the log table size under control. The higher the value, the more data is kept in the log tables. Try to estimate the number of transactions occurring in all the source tables during a retention period and be sure that the database and table space have enough storage capacity for all those transactions. The DBReplicator (engine) cleans up the log tables periodically, based on the retention setting in the connection dialog. If the engine is not running, the log tables are not cleaned up. This might create space problems in the database as the logs grow in size. If you stop the engine and you are not planning to run it again, be sure to remove all the mirroring synchronization replications.

In addition, if you have many table replications in a single group, using a single connection, all the replications share a master log table. Access to the log table for each source table can become a bottleneck if there are many transactions using the same master log and log tables. Syniti Replicate may report errors about locked tables during replication. Although Syniti Replicate is able to recover from these errors and continue replicating, a better approach is to prevent the errors by splitting the replications into multiple groups with multiple connections and multiple master log tables. First, create multiple source connections to the database. Use the Transaction Log Type field in the Connection Properties of each connection to open the Setup Info dialog and create a new master log table for each connection.

During replication:

- When a record is inserted in the source table, the INSERT trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains all the original values of the INSERT statement.
- When a record is deleted from a table, the DELETE trigger fires and inserts one record in the master table and one record in the log table associated with the source table. The record inserted in the log table contains the key values of the deleted record.

Syniti Replicate

- When a record is updated, the UPDATE trigger fires and inserts one record in the master table and two records in the log table associated with the source table. The two records inserted in the log table contain all the record values before and all the records after the update.

A transaction ID is saved both on the master records and log record to maintain a link between the transaction and the data changes for that transaction. See "Syniti Replicate Log Tables" for more details on the structure of the Master and Log tables.

Your system administrator needs to create and define appropriate table spaces and databases to hold the log tables. They should be large enough to handle the expected amount of replication data.

Syniti Replicate Log Tables

Log tables are used to record all data changes made to the source tables. They are populated by triggers that are fired when source tables are modified. Currently, log tables must reside in the replication source database. Note that log tables are created by Syniti Replicate only if they do not already exist in the database.

There are two log tables associated with each replication: a master table, common to all replications using that connection, and a log table, one for each replication. The master table keeps track of all the transactions affecting the source tables and it records general transactional information.

Master table structure

- `TID DECIMAL(31,0) GENERATED BY DEFAULT AS IDENTITY:`
  Transaction ID number associated with each record data change (transaction)

- `SNAME VARCHAR(128):`
  Name of the schema the transaction was applied to.

- `TNAME VARCHAR(128):`
  Name of the table the transaction was applied to.

- `TTS TIMESTAMP:`
  Transaction timestamp indicating when the transaction was submitted to the system

- `TUSER VARCHAR(128):`
  Name of the user who executed the transaction

Log Table Structure

The Log table contains the actual data changes for a specific source table. Its structure depends on the structure of the source table.

- `__TID DECIMAL(31,0):`

  Transaction ID, link to the corresponding record in the master table

- `__OP CHAR:`

  Indicates the type of operation:

  'I' INSERT
  'D' DELETE
  'B' UPDATE: values before the update

'A' UPDATE: values after the update

- `<Field list>`:

    All the columns of the source table with their original data type. For example if the source table was created as:

    ```
    CREATE TABLE SOURCET
    (ID INTEGER,
     FNAME VARCHAR(30),
     LNAME VARCHAR(50))
    ```

    The log table will have the following structure:

    ```
    CREATE TABLE _DBM__SOURCET
    (__TID DECIMAL(31,0),
     __OP  CHAR,
     ID    INTEGER,
     FNAME VARCHAR(30),
     LNAME VARCHAR(50))
    ```

Reading From/Managing Log Table

Syniti Replicate reads the log tables using the SELECT SQL statement. First, it queries the master table to see if new transactions came in since the last processed __TID. If transactions are found, Syniti Replicate queries the corresponding log table to collect the data changes and apply them to the target table.

The SELECTs on the master and log tables are sorted by the unique column __TID which ensures that all records will be read in the order that they were written. Syniti Replicate also uses the unique __TID column to keep track of the point where the last record was read and processed from the log tables.

SYNITI REPLICATE provides options to manage log records that have been read and replicated. They can be deleted from the log table as soon as they are processed or a retention time can be set to leave this record in the log tables for the specified number of hours.

Add a Source Connection Wizard

Select Provider Screen

Assembly Field

Locate the file IBM.Data.Informix.dll in the installation folder for the Informix .NET Provider.

Set Connection String Screen

Host

Type the IP address of your Informix server.

Server

Type the name or alias of the instance of the Informix server.

Service

The service name or port number through which the server is listening for connection requests.

Syniti Replicate

Database

The name of the database to which to connect within the server instance.

Protocol

A drop-down list with the default value of olsoctcp. Change the value according to the protocol used between the Informix .NET Data Provider and the database server.

Enable Transactional Replication Wizard

For transactional replications (mirroring and synchronization), use the Enable Transactional Replication wizard after setting up a source connection.

Log Type Screen

Select whether you plan to perform replications using the Log Server Agent or Triggers. The Setup Notes for Log Server Agent Transactional Replications are available in the Help Center.

For help completing the fields for trigger settings, use the Help button in the wizard.